

# quanteda

September 10, 2014

## R topics documented:

bigrams . . . . .	2
collocations . . . . .	3
corpus . . . . .	4
countSyllables . . . . .	5
describeTexts . . . . .	6
dfm . . . . .	6
dfm2ldaformat . . . . .	8
dfm2tmformat . . . . .	9
dfmSort . . . . .	9
dfmTrim . . . . .	10
directory . . . . .	11
docnames . . . . .	11
flatten.dictionary . . . . .	12
getRootFileNames . . . . .	13
getTextDir . . . . .	13
getTextDirGui . . . . .	14
getTextFiles . . . . .	15
getWordStat . . . . .	16
getWordStatCSV . . . . .	16
inaugCorpus . . . . .	16
inaugTexts . . . . .	17
kwic . . . . .	17
kwic2 . . . . .	18
likelihood.test . . . . .	19
MCMCirtPoisson1d . . . . .	19
metacorpus . . . . .	21
metacorpus<- . . . . .	22
naiveBayesText . . . . .	22
ngrams . . . . .	23
predict.naivebayes . . . . .	24
quantedaRefresh . . . . .	25
readWStatDict . . . . .	25
sample.corpus . . . . .	26

selectFeatures . . . . .	26
sentenceSeg . . . . .	28
settings . . . . .	28
stopwordsGet . . . . .	29
stopwordsRemove . . . . .	29
subset.corpus . . . . .	30
summary.corpus . . . . .	31
tagPos . . . . .	32
texts . . . . .	32
tf . . . . .	33
tfidf . . . . .	34
tokenise . . . . .	34
translate . . . . .	35
translate.corpus . . . . .	36
twitterSearch . . . . .	36
twitterStreamer . . . . .	37
twitterTerms . . . . .	37
uk2010immig . . . . .	38
wordcloudDfm . . . . .	38
wordfishMCMC . . . . .	39

<b>Index</b>	<b>42</b>
--------------	-----------

---

bigrams	<i>Create bigrams</i>
---------	-----------------------

---

**Description**

Create bigrams

**Usage**

```
bigrams(text, window = 1, concatenator = "_", include.unigrams = FALSE,
...)
```

**Arguments**

text	character vector containing the texts from which bigrams will be constructed
window	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words. This is only available for bigrams, not for <a href="#">ngram</a> .
concatenator	character for combining words, default is _ (underscore) character
include.unigrams	if TRUE, return unigrams as well
...	additional arguments passed to <a href="#">tokenize</a>

**Value**

a character vector of bigrams

**Author(s)**

Kohei Watanabe and Ken Benoit

**Examples**

```
bigrams("The quick brown fox jumped over the lazy dog.")
bigrams("The quick brown fox jumped over the lazy dog.", window=2)
```

---

collocations

*Detect collocations in a text*

---

**Description**

returns a list of collocations. Note: Currently works only for pairs (bigram collocations).

**Usage**

```
collocations(text = NULL, file = NULL, top = NA, distance = 2, n = 2,
  method = c("lr", "chi2", "mi"))
```

**Arguments**

text	a text or vector of texts
file	a filename containing a text
top	threshold number for number of collocations to be returned (in descending order of association value)
distance	distance between pairs of collocations
method	association measure for detecting collocations
n	Only bigrams (n=2) implemented so far.

**Value**

A list of collocations, their frequencies, and their test statistics

**Author(s)**

Kenneth Benoit

**Examples**

```
data(inaugCorpus)
collocations(texts(inaugCorpus)[1], top=50)
collocations(texts(inaugCorpus)[1], top=50, method="chi2")
```

---

corpus	<i>Constructor for corpus objects</i>
--------	---------------------------------------

---

## Description

Creates a corpus from a character vector (of texts), or an object such as a directory containing text files. Corpus-level meta-data can be specified at creation, containing (for example) citation information and notes.

## Usage

```
corpus(texts, ...)

## Default S3 method:
corpus()

## S3 method for class 'directory'
corpus(path, enc = NULL, docnames = NULL,
        docvarsfrom = c("filenames", "headers"), docvarnames = NULL, sep = "_",
        source = NULL, notes = NULL, citation = NULL)

## S3 method for class 'character'
corpus(texts, enc = NULL, docnames = NULL,
        docvars = NULL, source = NULL, notes = NULL, citation = NULL, ...)
```

## Arguments

texts	A character vector of texts, or a filepath to a directory containing text documents.
docvarsfrom	Argument to specify where docvars are to be taken, from parsing the filenames ( <a href="#">filenames</a> ) separated by sep or from meta-data embedded in the text file header (headers).
docvarnames	Character vector of variable names for docvars
sep	Separator if <a href="#">docvar</a> names are taken from the filenames.
texts	A character vector containing the texts
docnames	Names to be assigned to the texts, defaults to the names of the character vector (if any), otherwise assigns "text1", "text2", etc.
docvars	A data frame of attributes that is associated with each text.
source	A string specifying the source of the texts, used for referencing.
notes	A string containing notes about who created the text, warnings, To Dos, etc.

**Examples**

```
## Not run: corpus() # pop up a file selection GUI interface

## Not run:
tempcorpus <- corpus(directory("~/Dropbox/QUANTESS/corpora/ukManRenamed"), enc="UTF-8", source="Ken's manifesto")
summary(tempcorpus, n=10)

## End(Not run)
corpus(inaugTexts)
uk2010immigCorpus <- corpus(uk2010immig, docvars=data.frame(party=names(uk2010immig)), enc="UTF-8")
```

---

countSyllables	<i>Returns a count of the number of syllables in the input This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.</i>
----------------	--

---

**Description**

Returns a count of the number of syllables in the input This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.

**Usage**

```
countSyllables(sourceText)
```

**Arguments**

sourceText      Character vector of texts whose syllables will be counted

**Details**

This only works for English.

**Value**

numeric Named vector of counts of the number of syllables for each element of sourceText. When a word is not available in the lookup table, its syllables are estimated by counting the number of (English) vowels in the word.

**Examples**

```
countSyllables("This is an example sentence.")
myTexts <- c("Text one.", "Superduper text number two.", "One more for the road.")
names(myTexts) <- paste("myText", 1:3, sep="")
countSyllables(myTexts)
```

---

describeTexts	<i>print a summary of texts Prints to the console a description of the texts, including number of types, tokens, and sentences</i>
---------------	--

---

### Description

print a summary of texts Prints to the console a description of the texts, including number of types, tokens, and sentences

### Usage

```
describeTexts(txts, verbose = TRUE)
```

### Arguments

txts	The texts to be described
verbose	Default is TRUE. Set to false to suppress output messages

### Examples

```
describeTexts(c("testing this text", "and this one"))
describeTexts(uk2010immig)
```

---

dfm	<i>Create a document-feature matrix from a corpus object</i>
-----	--

---

### Description

returns a document by feature matrix compatible with austin. A typical usage would be to produce a word-frequency matrix where the cells are counts of words by document.

### Usage

```
dfm(corp, feature = c("word"), stem = FALSE, stopwords = NULL,
    bigram = FALSE, groups = NULL, subset = NULL, verbose = TRUE,
    dictionary = NULL, dictionary_regex = FALSE, addto = NULL)

## S3 method for class 'corpus'
dfm(corp, feature = c("word"), stem = FALSE,
    stopwords = NULL, bigram = FALSE, groups = NULL, subset = NULL,
    verbose = TRUE, dictionary = NULL, dictionary_regex = FALSE,
    addto = NULL)

## S3 method for class 'character'
dfm(textvec, feature = c("word"), stem = FALSE,
    stopwords = NULL, bigram = FALSE, verbose = TRUE, dictionary = NULL,
    dictionary_regex = FALSE, addto = NULL)
```

**Arguments**

corp	Corpus from which to generate the document-feature matrix
feature	Feature to count (e.g. words)
stem	Stem the words
stopwords	A character vector of stopwords that will be removed from the text when constructing the <a href="#">dfm</a> . If NULL (default) then no stopwords will be applied. If "TRUE" then it currently defaults to <a href="#">stopwords</a> .
groups	Grouping variable for aggregating documents
subset	Expression for subsetting the corpus before processing
verbose	Get info to screen on the progress
dictionary	A list of character vector dictionary entries, including regular expressions (see examples)
dictionary_regex	TRUE means the dictionary is already in regular expression format, otherwise it will be converted from "wildcard" format
addto	NULL by default, but if an existing dfm object is specified, then the new dfm will be added to the one named. If both <a href="#">dfm</a> 's are built from dictionaries, the combined dfm will have its Non_Dictionary total adjusted.

**Value**

A matrix object with row names equal to the document names and column names equal to the feature labels. This matrix has `names(dimnames) = c("docs", "words")` to make it conformable to an [wfm](#) object.

**Author(s)**

Kenneth Benoit

**Examples**

```
data(inaugCorpus)
wfm <- dfm(inaugCorpus)

## by president, after 1960
wfmByPres2010 <- dfm(subset(inaugCorpus, year>1960), groups="president")

## with dictionaries
mycorpus <- subset(inaugCorpus, year>1960)
mydict <- list(christmas=c("Christmas", "Santa", "holiday"),
               opposition=c("Opposition", "reject", "notincorpus"),
               taxing="taxing",
               taxation="taxation",
               taxregex="tax*")
dictDfm <- dfm(mycorpus, dictionary=mydict)
dictDfm
```

```
## removing stopwords
testText <- "The quick brown fox named Seamus jumps over the lazy dog Rory, with Tom's newspaper in his mouth."#
testCorpus <- corpus(testText)
dfm(testCorpus, stopwords=TRUE)
if (require(tm)) {
}
```

---

dfm2ldaformat	<i>Convert a quanteda <a href="#">dfm</a> (document feature matrix) into a the data format needed by <a href="#">lda</a></i>
---------------	--

---

## Description

Convert a quanteda [dfm](#) (document feature matrix) into a the data format needed by [lda](#)

## Usage

```
dfm2ldaformat(d)
```

## Arguments

d                    A [dfm](#) object

## Value

A list with components "documents" and "vocab" as needed by [lda.collapsed.gibbs.sampler](#)

## Examples

```
data(inaugCorpus)
inaugCorpus <- subset(inaugCorpus, year>1960)
# create document-feature matrix, remove stopwords
d <- dfm(inaugCorpus, stopwords=TRUE)
# trim low frequency words
d <- dfmTrim(d, minCount=5, minDoc=3)
td <- dfm2ldaformat(d)
if (require(lda)) {
  tmodel.lda <- result <- lda.collapsed.gibbs.sampler(documents=td$documents,
                                                    K=10,
                                                    vocab=td$vocab,
                                                    num.iterations=50, alpha=0.1, eta=0.1)
}
top.topic.words(tmodel.lda$topics, 10, by.score=TRUE) # top five words in each topic
```



---

dfm2tmformat	<i>Convert a quanteda <a href="#">dfm</a> (document feature matrix) into a <b>tm</b> <a href="#">DocumentTermMatrix</a></i>
--------------	---

---

## Description

**tm** represents sparse document-feature matrixes in the [simple triplet matrix](#) format of the package **slam**. This function converts a [dfm](#) into a [DocumentTermMatrix](#), for working with the [dfm](#) in **tm** or in other packages that expect this format, such as **topicmodels**.

## Usage

```
dfm2tmformat(d, weighting = weightTf, ...)
```

## Arguments

d	A <a href="#">dfm</a> object
weighting	<b>tm</b> 's coercion function accepts weightings such as tf-idf, see <b>tm</b> 's <a href="#">as.DocumentTermMatrix</a> for a list of possible arguments. The default is just tf (term frequency)

## Value

A simple triplet matrix of class [as.DocumentTermMatrix](#)

## Examples

```
data(inaugCorpus)
inaugCorpus <- subset(inaugCorpus, year==2010)
d <- dfmTrim(dfm(inaugCorpus), minCount=5, minDoc=3)
dim(d)
td <- dfm2tmformat(d)
length(td$v)
if (require(topicmodels)) tmodel.lda <- LDA(td, control = list(alpha = 0.1), k = 4)
```

---

dfmSort	<i>sort a <a href="#">dfm</a> by one or more margins</i>
---------	--

---

## Description

Sorts a [dfm](#) by documents or words

## Usage

```
dfmSort(x, margin = c("words", "docs", "both"), decreasing = TRUE)
```

**Arguments**

dfm	Document-feature matrix created by <a href="#">dfm</a>
margin	which margin to sort on words to sort words, docs to sort documents, and both to sort both
decreasing	TRUE (default) if sort will be in descending order

**Value**

A sorted [dfm](#) matrix object

**Author(s)**

Ken Benoit

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[, 1:10]
dtm <- dfmSort(dtm, "words")
dfmSort(dtm)[, 1:10]
dfmSort(dtm, "both")[, 1:10]
```

---

dfmTrim

*Trim a dfm based on a subset of features and words*


---

**Description**

Returns a document by feature matrix reduced in size based on document and term frequency, and/or subsampling.

**Usage**

```
dfmTrim(dfm, minCount = 5, minDoc = 5, sample = NULL, verbose = TRUE)
```

**Arguments**

dfm	Document-feature matrix created by <a href="#">dfm</a>
minCount	minimum feature count
minDoc	minimum number of documents in which a feature appears
sample	how many features to retain (based on random selection)
verbose	print messages

**Value**

A dfm matrix object reduced in size.

**Author(s)**

Will Lowe, adapted by Ken Benoit

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dim(dtm)
dtmReduced <- dfmTrim(dtm, minCount=10, minDoc=2) # only words occurring at least 5 times and in at least 2 documents
dim(dtmReduced)
dtmSampled <- dfmTrim(dtm, sample=50) # top 200 words
dim(dtmSampled) # 196 x 200 words
```

---

directory

*Function to declare a connection to a directory (containing files)*

---

**Description**

Function to declare a connection to a directory, although unlike [file](#) it does not require closing. If the directory does not exist, the function will return an error.

**Usage**

```
directory(path)
```

**Arguments**

path                      String describing the full path of the directory

**Examples**

```
## Not run: mydir <- directory("~/Dropbox/QUANTESS/corpora/ukManRenamed")
corpus(mydir)
## End(Not run)
```

---

docnames

#

---

**Description**

#

**Usage**

```
docnames(corp)
```

---

flatten.dictionary	<i>Flatten a hierarchical dictionary into a list of character vectors</i>
--------------------	---

---

### Description

Converts a hierarchical dictionary (a named list of named lists, ending in character vectors at the lowest level) into a flat list of character vectors. Works like `unlist(dictionary, recursive=TRUE)` except that the recursion does not go to the bottom level.

### Usage

```
flatten.dictionary(elms, parent = "", dict = list())
```

### Arguments

elms	list to be flattened
parent	parent list name, gets built up through recursion in the same way that <code>unlist(dictionary, recursive=TRUE)</code> works
dict	the bottom list of dictionary entries ("synonyms") passed up from recursive calls

### Details

Called by `dfm()`

### Value

A dictionary flattened down one level further than the one passed

### Author(s)

Kohei Watanabe

### Examples

```
dictPopulismEN <-
  list(populism=c("elit*", "consensus*", "undemocratic*", "referend*",
    "corrupt*", "propagand", "politici*", "*deceit*",
    "*deceiv*", "*betray*", "shame*", "scandal*", "truth*",
    "dishonest*", "establishm*", "ruling*"))
flatten.dictionary(dictPopulismEN)

hdict <- list(level1a = list(level1a1 = c("l1a11", "l1a12"),
  level1a2 = c("l1a21", "l1a22")),
  level1b = list(level1b1 = c("l1b11", "l1b12"),
    level1b2 = c("l1b21", "l1b22", "l1b23")),
  level1c = list(level1c1a = list(level1c1a1 = c("lowest1", "lowest2")),
    level1c1b = list(level1c1b1 = c("lowestalone"))))
flatten.dictionary(hdict)
```

---

getRootFileNames	<i>Truncate absolute filepaths to root filenames</i>
------------------	--

---

**Description**

This function takes an absolute filepath and returns just the document name

**Usage**

```
getRootFileNames(longFilenames)
```

**Arguments**

longFilenames    Absolute filenames including a full path with directory

**Value**

character vector of filenames withouth directory path

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:  
getRootFileNames('/home/paul/documents/libdem09.txt')  
  
## End(Not run)
```

---

getTextDir	<i>loads all text files from a given directory</i>
------------	--

---

**Description**

given a directory name, get a list of all files in that directory and load them into a character vector using getTextFiles

**Usage**

```
getTextDir(dirname, enc = "detect")
```

**Arguments**

dirname            A directory path

**Value**

character vector of texts read from disk

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:  
getTextDir('/home/paul/documents/')  
  
## End(Not run)
```

---

getTextDirGui	<i>provides a gui interface to choose a gui to load texts from</i>
---------------	--

---

**Description**

launches a GUI to allow the user to choose a directory from which to load all files.

**Usage**

```
getTextDirGui()
```

**Value**

character vector of texts read from disk

**Author(s)**

Paul Nulty

**Examples**

```
## Not run:  
getTextFiles('/home/paul/documents/libdem09.txt')  
  
## End(Not run)
```

---

getTextFiles	<i>load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts</i>
--------------	--

---

### Description

load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts

### Usage

```
getTextFiles(filenames, textnames = NULL, enc = "unknown",  
             verbose = FALSE)
```

### Arguments

filenames	a vector of paths to text files
textnames	names to assign to the texts
verbose	If TRUE, print out names of files being read. Default is FALSE

### Value

character vector of texts read from disk

### Author(s)

Paul Nulty

### Examples

```
## Not run:  
getTextFiles('/home/paul/documents/libdem09.txt')  
  
## End(Not run)
```

---

getWordStat	<i>Imports a Wordstat corpus from an XML file</i>
-------------	---

---

**Description**

Reads in a wordstat XML file and creates a corpus object with the document as text and variables as attributes

**Usage**

```
getWordStat(filename = NULL)
```

**Arguments**

filename	Path to wordstat XML file
----------	---------------------------

---

getWordStatCSV	<i>Imports a Wordstat corpus from a CSV file</i>
----------------	--

---

**Description**

Reads in a wordstat CSV file and creates a corpus object with the document as text and variables as attributes

**Usage**

```
getWordStatCSV(filename = NULL)
```

**Arguments**

filename	Path to wordstat CSV file
----------	---------------------------

---

inaugCorpus	<i>Corpus of US presidential inauguration speeches</i>
-------------	--

---

**Description**

Corpus of US presidential inauguration speeches



---

inaugTexts

*Texts of US presidential inauguration speeches*


---

### Description

Character vector of US presidential inauguration speeches

---

kwic

*List key words in context from a text or a corpus of texts.*


---

### Description

For a text or a collection of texts (in a quanteda corpus object), return a list of a keyword supplied by the user in its immediate context, identifying the source text and the word index number within the source text. (Not the line number, since the text may or may not be segmented using end-of-line delimiters.)

### Usage

```
kwic(text, word, window = 5, regex = TRUE)

## S3 method for class 'character'
kwic(texts, word, window = 5, regex = TRUE)

## S3 method for class 'corpus'
kwic(corp, word, window = 5, regex = TRUE)
```

### Arguments

text	A text character scalar or a quanteda corpus. (Currently does not support character vectors.)
word	A keyword chosen by the user.
window	The number of context words to be displayed around the keyword.
regex	If TRUE (default), then "word" is a regular expression, otherwise only match the whole word. Note that if regex=TRUE and no special regular expression characters are used in the search query, then the concordance will include all words in which the search term appears, and not just when it appears as an entire word. (For instance, searching for the word "key" will also return "whiskey".)
texts	a vector of texts
corp	a quanteda corpus object

**Value**

A data frame with the context before (preword), the keyword in its original format (word, preserving case and attached punctuation), and the context after (postword). The rows of the dataframe will be named with the word index position, or the text name and the index position for a corpus object.

**Author(s)**

Kenneth Benoit and Paul Nulty

**Examples**

```
kwic(inaugTexts, "terror")
kwic(inaugTexts, "terror", regex=FALSE) # returns only whole word, without trailing punctuation
data(iebudgets)
kwic(subset(iebudgets, year==2010), "Christmas", window=4) # on a corpus
```

---

kwic2

*This function is an alternative KWIC*

---

**Description**

This function is an alternative KWIC

**Usage**

```
kwic2(texts, word, window = 30, filter = "", location = TRUE,
      case = TRUE)
```

**Arguments**

text	Texts
word	Word of interest
window	Window span in character
filter	Filter files in texts by regular expression
location	Show location of the word
case	Ignore case

**Value**

cfvm2 Collocatons as data frame

**Author(s)**

Kohei Watanabent

**Examples**

```
## Not run:
kwc2(texts, "we", filter = '_2010', location=TRUE)

## End(Not run)
```

---

likelihood.test	<i>likelihood test for 2x2 tables</i>
-----------------	---------------------------------------

---

**Description**

returns a list of values

**Usage**

```
likelihood.test(x)
```

**Arguments**

x                      a contingency table or matrix object

**Value**

A list of return values

**Author(s)**

Kenneth Benoit

---

MCMCirtPoisson1d	<i>Bayesian-MCMC version of a 1-dimensional Poisson IRT scaling model</i>
------------------	---

---

**Description**

MCMCirtPoisson1d implements a flexible, Bayesian model estimated in JAGS using MCMC. It is based on the implementation of [wordfish](#) from the [austin](#) package. Options include specifying a model for alpha using document-level covariates, and partitioning the word parameters into different subsets, for instance, countries.

**Usage**

```
MCMCirtPoisson1d(dtm, dir = c(1, 2), control = list(sigma = 3, startparams =
  NULL), verbose = TRUE, itembase = 1, startRandom = FALSE, nChains = 1,
  nAdapt = 100, nUpdate = 300, nSamples = 200, nThin = 1, ...)
```

**Arguments**

<code>dtm</code>	The document-term matrix. Ideally, documents form the rows of this matrix and words the columns, although it should be correctly coerced into the correct shape.
<code>dir</code>	A two-element vector, enforcing direction constraints on theta and beta, which ensure that $\text{theta}[\text{dir}[1]] < \text{theta}[\text{dir}[2]]$ . The elements of <code>dir</code> will index documents.
<code>control</code>	list specifies options for the estimation process. These are: <code>tol</code> , the proportional change in log likelihood sufficient to halt estimation, <code>sigma</code> the standard deviation for the beta prior in poisson form, and <code>startparams</code> a previously fitted wordfish model. <code>verbose</code> generates a running commentary during estimation. See <a href="#">wordfish</a> .
<code>itembase</code>	A index or column name from <code>dtm</code> indicating which item should be used as the reference category. (These will have $\beta_j = 0$ and $\alpha_j = 0$ .) The default is 1, to use the first category. If set to NULL then no constraints will be implemented. See details.
<code>verbose</code>	Turn this on for messages. Default is TRUE.
<code>startRandom</code>	FALSE by default, uses random starting values (good for multiple chains) if TRUE
<code>nChains</code>	Number of chains to run in JAGS.
<code>nAdapt</code>	Adaptation iterations in JAGS.
<code>nUpdate</code>	Update iterations in JAGS.
<code>nSamples</code>	Number of posterior samples to draw in JAGS.
<code>nThin</code>	Thinning parameter for drawing posterior samples in JAGS.
<code>...</code>	Additional arguments passed through.

**Details**

The ability to constrain an item is designed to make the additive Poisson GLM mathematically equivalent to the multinomial model for  $R \times C$  contingency tables. We recommend setting a neutral category to have  $\psi_0 = 0$  and  $\beta_0 = 0$ , for example the word "the" for a text count model (assuming this word has not been removed). Note: Currently the item-level return values will be returned in the original order supplied (`psi` and `beta`) but this is not true yet for the `mcmc.samples` value, which will have the constrained category as index 1. (We will fix this soon.)

**Value**

An augmented [wordfish](#) class object with additional stuff packed in. To be documented.

**Author(s)**

Kenneth Benoit

**Examples**

```
## Not run:
data(iebudgets)
# extract just the 2010 debates
iebudgets2010 <- subset(iebudgets, year==2010)

# create a document-term matrix and set the word margin to the columns
dtm <- dfm(iebudgets2010)

# estimate the maximum likelihood wordfish model from austin
require(austin)
iebudgets2010_wordfish <- wordfish(as.wfm(dtm, word.margin=2), dir=c(2,1))

# estimate the MCMC model, default values
iebudgets2010_wordfishMCMC <- MCMCirtPoisson1d(dtm, itembase="the", dir=c(2,1))
iebudgets2010_wordfishMCMC_unconstrained <- MCMCirtPoisson1d(dtm, dir=c(2,1))

# compare the estimates of  $\theta_i$ 
require(psych)
pairs.panels(data.frame(ML=iebudgets2010_wordfish$theta,
                        PoissonThe=iebudgets2010_wordfishMCMC$theta,
                        PoissonUnconst=iebudgets2010_wordfishMCMC_unconstrained$theta),
             smooth=FALSE, scale=FALSE, ellipses=FALSE, lm=TRUE, cex.cor=2.5)
# inspect a known "opposition" word beta values
iebudgets2010_wordfish$beta[which(iebudgets2010_wordfishMCMC_unconstrained$words=="fianna")]
iebudgets2010_wordfishMCMC$beta[which(iebudgets2010_wordfishMCMC_unconstrained$words=="fianna")]
iebudgets2010_wordfishMCMC_unconstrained$beta[which(iebudgets2010_wordfishMCMC_unconstrained$words=="fianna")]

# random starting values, for three chains
dtm.sample <- trim(dtm, sample=200)
iebudgets2010_wordfishMCMC_sample <- MCMCirtPoisson1d(dtm.sample, dir=c(2,1), startRandom=TRUE, nChains=3)

## End(Not run)
```

---

metacorpus

*Corpus-level metadata*


---

**Description**

Get or set the corpus-level metadata in a quanteda corpus object.

**Usage**

```
metacorpus(corp, fields = NULL)
```

```
metacorpus(corp, fields) <- value
```

**Arguments**

corp	A quanteda corpus object
fields	Metadata field names. If NULL (default), return all metadata names.

**Value**

For metacorporus, a list of the metadata fields in the corpus.  
 For metacorporus <-, the corpus with the updated metadata.

**Examples**

```
metacorporus(inaugTexts)
metacorporus(inaugTexts, "source")
metacorporus(inaugTexts, "citation") <- "Presidential Speeches Online Project (2014)."
```

---

metacorporus<-	<i>replacement function for corpus-level data</i>
----------------	---

---

**Description**

replacement function for corpus-level data

**Usage**

```
metacorporus(corp, fields) <- value
```

---

naiveBayesText	<i>Naive Bayes classifier for texts</i>
----------------	---

---

**Description**

Naive Bayes classifier for texts

**Usage**

```
naiveBayesText(x, y, smooth = 1, prior = "uniform",
  distribution = "multinomial", ...)
```

**Arguments**

x	character vector of training texts
y	character vector of test texts
smooth	smoothing parameter for feature counts by class
prior	prior distribution on texts, see details
distribution	count model for text features, can be multinomial or Bernoulli
...	

**Details**

Currently working for vectors of texts.

**Value**

A list of return values, consisting of:

call	original function call
PwGc	probability of the word given the class (empirical likelihood)
Pc	class prior probability
PcGw	posterior class probability given the word
Pw	baseline probability of the word
data	list consisting of x training class, and y test class
distribution	the distribution argument
prior	argument passed as a prior
smooth	smoothing parameter

**Author(s)**

Kenneth Benoit

---

ngrams

*Create ngrams*

---

**Description**

Create a set of ngrams (words in sequence) from a text.

**Usage**

```
ngrams(text, n = 2, concatenator = "_", include.all = FALSE, ...)
```

**Arguments**

text	character vector containing the texts from which ngrams will be extracted
n	the number of tokens to concatenate. Default is 2 for bigrams.
window	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words.
concatenator	character for combining words, default is _ (underscore) character
include.all	if TRUE, add n-1...1 grams to the returned list
...	additional arguments passed to <a href="#">tokenize</a>

**Value**

a character vector of ngrams

**Author(s)**

Ken Benoit, Kohei Watanabe, Paul Nulty

**Examples**

```
ngrams("The quick brown fox jumped over the lazy dog.", n=2)
ngrams("The quick brown fox jumped over the lazy dog.", n=3)
ngrams("The quick brown fox jumped over the lazy dog.", n=3, concatenator="~")
ngrams("The quick brown fox jumped over the lazy dog.", n=3, include.all=TRUE)
```

---

predict.naivebayes	<i>prediction method for Naive Bayes classifiers</i>
--------------------	--

---

**Description**

prediction method for Naive Bayes classifier objects

**Usage**

```
## S3 method for class 'naivebayes'
predict(object, newdata = NULL, scores = c(-1, 1))
```

**Arguments**

object	a naivebayes class object
newdata	new data on which to perform classification
scores	"reference" values when the wordscores equivalent implementation of Naive Bayes prediction is used. Default is c(-1, 1).

**Details**

implements class predictions using trained Naive Bayes examples (from naiveBayesText())

**Value**

A list of two data frames, named docs and words corresponding to word- and document-level predicted quantities

docs	data frame with document-level predictive quantities: nb.predicted, ws.predicted, bs.predicted, PcGw, wordscore.doc, bayesscore.doc, posterior.diff, posterior.logdiff. Note that the diff quantities are currently implemented only for two-class solutions.
words	data-frame with word-level predictive quantities: wordscore.word, bayesscore.word



## Author(s)

Kenneth Benoit

---

quantedaRefresh	<i>Re-install <a href="#">quanteda</a> from github</i>
-----------------	--

---

## Description

Refresh the installation from the github repository for the package. Useful if you need to pull the latest changes.

## Usage

```
quantedaRefresh(branch = c("dev", "master"))
```

## Arguments

branch	default is "dev"
--------	------------------

## Value

Nothing

## Author(s)

Kenneth Benoit

---

readWStatDict	<i>Make a flattened list from a hierarchical wordstat dictionary</i>
---------------	--

---

## Description

Make a flattened list from a hierarchical wordstat dictionary

## Usage

```
readWStatDict(path)
```

## Arguments

path	path to the wordstat dictionary file
------	--------------------------------------

## Value

flattened dictionary as a list

---

sample.corpus	<i>Corpus sampling</i>
---------------	------------------------

---

### Description

Takes a random sample of the specified size from a corpus, with or without replacement

### Usage

```
sample.corpus(corpus, size = n, replace = FALSE, prob = NULL)
```

### Arguments

corpus	An existing corpus to be sampled
size	A positive number, the number of texts to return
replace	Should sampling be with replacement?
prob	Not implemented

### Examples

```
data(inaugCorpus)
inaugSamp <- sample(inaugCorpus, 200, replace=TRUE)
```

---

selectFeatures	<i>extract feature words This function takes type of feature extractor and a word frequency matrix with binary class (1/0) to select features in class one. 'wsll' and 'wschisq' replicates of 'Keyness' of Wordsmith Tools.</i>
----------------	--

---

### Description

extract feature words This function takes type of feature extractor and a word frequency matrix with binary class (1/0) to select features in class one. 'wsll' and 'wschisq' replicates of 'Keyness' of Wordsmith Tools.

extract feature words This function takes type of feature extractor and a word frequency matrix with binary class (1/0) to select features in class one. 'wsll' and 'wschisq' replicates of 'Keyness' of Wordsmith Tools.

### Usage

```
selectFeatures(extractor, dfm, class, smooth = 1, show = 10)
```

```
selectFeatures(extractor, dfm, class, smooth = 1, show = 10)
```

**Arguments**

extractor	Type of feature extractor
dfm	Word frequency matrix
class	Biarny class
smooth	Smoothing constant
show	Number of features shown
extractor	Type of feature extractor
dfm	Word frequency matrix
class	Biarny class
smooth	Smoothing constant
show	Number of features shown

**Value**

data frame of feature words  
data frame of feature words

**Author(s)**

Kohei Watanabe  
Kohei Watanabe

**Examples**

```
## Not run:
texts <- getTextDir("/home/kohei/Documents/budget_2010/")
class <- rep(0, length(texts))
class[grepl("_LAB", names(texts))] <- 1
class[grepl("_FF", names(texts))] <- 0
corpus <- corpusCreate(texts, attribs=list(class=class))
dfm <- dfm(corpus)
features <- selectFeatures('ll', dfm, corpus$attribs$class, smooth=1)

## End(Not run)
## Not run:
texts <- getTextDir("/home/kohei/Documents/budget_2010/")
class <- rep(0, length(texts))
class[grepl("_LAB", names(texts))] <- 1
class[grepl("_FF", names(texts))] <- 0
corpus <- corpusCreate(texts, attribs=list(class=class))
dfm <- dfm(corpus)
features <- selectFeatures('ll', dfm, corpus$attribs$class, smooth=1)

## End(Not run)
```



---

stopwordsGet	<i>access stopwords</i>
--------------	-------------------------

---

### Description

This function retrieves stopwords from the type specified in the `kind` argument and returns the stopword list as a character vector. The default is English.

### Usage

```
stopwordsGet(kind = "english")
```

### Arguments

<code>kind</code>	The pre-set kind of stopwords (as a character string)
-------------------	---

### Value

a character vector or dfm with stopwords removed

### Examples

```
stopwordsGet()
stopwordsGet("italian")
```

---

stopwordsRemove	<i>remove stopwords from a text or dfm</i>
-----------------	--

---

### Description

This function takes a character vector or dfm and removes words in the `remove` common or 'semantically empty' words from a text.

### Usage

```
stopwordsRemove(text, stopwords = NULL)

## S3 method for class 'character'
stopwordsRemove(text, stopwords = NULL)

## S3 method for class 'matrix'
stopwordsRemove(text, stopwords = NULL)
```

### Arguments

<code>text</code>	Text from which stopwords will be removed
<code>stopwords</code>	Character vector of stopwords to remove

**Details**

This function takes a character vector 'text' and removes words in the list provided in 'stopwords'. If no list of stopwords is provided a default list for English is used.

**Value**

a character vector or dfm with stopwords removed

**Examples**

```
## examples for character objects
someText <- "Here is an example of text containing some stopwords we want to remove."
itText <- "Ecco un esempio di testo contenente alcune parole non significative che vogliamo rimuovere."
stopwordsRemove(someText)
stopwordsRemove(someText, stopwordsGet("SMART"))
stopwordsRemove(itText, stopwordsGet("italian"))
stopwordsRemove(someText, c("containing", "example"))

## example for dfm objects
data(iebudgets)
wfm <- dfm(subset(iebudgets, year==2010))
wfm.nostopwords <- stopwordsRemove(wfm)
dim(wfm)
dim(wfm.nostopwords)
dim(stopwordsRemove(wfm, stopwordsGet("SMART")))
```

---

subset.corpus	<i>extract a subset of a corpus</i>
---------------	-------------------------------------

---

**Description**

Works just like the normal subset command but for corpus objects

**Usage**

```
## S3 method for class 'corpus'
subset(corpus, subset = NULL, select = NULL)
```

**Arguments**

corpus	corpus object to be subsetted.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	expression, indicating the attributes to select from the corpus

**Value**

corpus object

**Examples**

```
## Not run:
data(inaugCorpus)
inaugCorpus <- subset(inaugCorpus, year==2010)
summary(iebudgets2010)
iebudgetsCarter <- subset(iebudgets, speaker="Carter", select=c(speaker, year))
summary(iebudgetsLenihan)

## End(Not run)
```

---

summary.corpus	<i>Corpus summary</i>
----------------	-----------------------

---

**Description**

Displays information about a corpus object, including attributes and metadata such as date of number of texts, creation and source.

**Usage**

```
## S3 method for class 'corpus'
summary(corp, n = 100, verbose = TRUE, meta = FALSE)
```

**Arguments**

corp	corpus to be summarized
n	maximum number of texts to describe, default=100
verbose	FALSE to turn off printed output
meta	TRUE to include document-level meta-data

**Examples**

```
summary(inaugCorpus)
summary(inaugCorpus, n=10, printdocnames=FALSE)
mycorpus <- corpus(uk2010immig, docvars=data.frame(party=names(uk2010immig)), enc="UTF-8")
summary(mycorpus, meta=TRUE) # show the meta-data
mysummary <- summary(mycorpus, verbose=FALSE) # (quietly) assign the results
mysummary$Types / mysummary$Tokens           # crude type-token ratio
```

---

tagPos	<i>Returns a table of the occurrences of different parts of speech in a sentence This function takes a sentence and tags each word with it's part of speech using openNLP's POS tagger, then returns a table of the parts of speech</i>
--------	---

---

### Description

[http://www.ling.upenn.edu/courses/Fall\\_2003/ling001/penn\\_treebank\\_pos.html](http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

### Usage

```
tagPos(sentence)
```

### Arguments

sentence	Sentence to be tagged
----------	-----------------------

### Examples

```
## Not run:
tagPos("This is an example sentence with nouns and verbs for tagging.")

## End(Not run)
```

---

texts	<i>Corpus texts</i>
-------	---------------------

---

### Description

Get or replace the texts in a quanteda corpus object.

### Usage

```
texts(corp)

texts(corp, rownames = FALSE) <- value
```

### Arguments

corp	A quanteda corpus object
rownames	If TRUE, overwrite the names of the documents with names from assigned object.



**Value**

For `texts`, a character vector of the texts in the corpus. For `texts <-`, the corpus with the updated texts.

**Examples**

```
texts(inaugCorpus)[1]
texts(inaugTexts)[55] <- "GW Bush's second inaugural address, the condensed version."
```

---

tf	<i>normalizes the term frequencies a dfm</i>
----	--

---

**Description**

Returns a matrix of term weights, as a [dfm](#) object

**Usage**

```
tf(x)
```

**Arguments**

dfm	Document-feature matrix created by <a href="#">dfm</a>
-----	--

**Value**

A dfm matrix object where values are relative term proportions within the document

**Author(s)**

Ken Benoit

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tf(dtm)[1:10, 100:110]
```

---

tfidf	<i>compute the tf-idf weights of a dfm</i>
-------	--

---

**Description**

Returns a matrix of tf-idf weights, as a [dfm](#) object

**Usage**

```
tfidf(x, normalize = TRUE)
```

**Arguments**

dfm	Document-feature matrix created by <a href="#">dfm</a>
normalize	whether to normalize term frequency by document totals

**Value**

A dfm matrix object where values are tf-idf weights

**Author(s)**

Ken Benoit

**Examples**

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tfidf(dtm)[1:10, 100:110]
tfidf(dtm, normalize=FALSE)[1:10, 100:110]
```

---

tokenise	<i>tokenise a corpus and store the results</i>
----------	--

---

**Description**

Tokenize the corpus and store the results as a dfm, along with counts for the syllables of each token, the number of sentences per text, and the number of paragraphs per text.

**Usage**

```
tokenise(x, ...)

## S3 method for class 'corpus'
tokenise(corp)
```

**Arguments**

corp                      Corpus to be tokenized

**Value**

tokens                    A list consisting of the following:  
 \$dfm                     A [dfm](#) document-feature matrix object created with [settings](#).  
 \$nwords                  A vector of token counts for each document.  
 \$ntypes                  A vector of type counts for each document.  
 \$nsents                  A vector of sentence counts for each document.  
 \$nparagr                 A vector of paragraph counts for each document.

**Examples**

```
mycorpus <- corpus(uk2010immig)
mycorpus
tokenise(mycorpus)
mycorpus
```

---

translate	<i>Send text to the google translate research API This function translates a text by sending it to the google translate API.</i>
-----------	--

---

**Description**

Send text to the google translate research API This function translates a text by sending it to the google translate API.

**Usage**

```
translate(sourceText, sourceLanguage, targetLanguage, key = NULL,
  verbose = FALSE)
```

**Arguments**

sourceText              Text to be translated  
 sourceLanguage        Language of the source text  
 targetLanguage        Language of the translated text  
 key                     API key for Google Translate research API

**Examples**

```
## Not run: translation <- translate(original, fr, de, key='insertkeyhere')
```

---

translate.corpus	<i>Send a corpus to the google translate research API This function translates a the texts in a corpus by sending them to the google translate API.</i>
------------------	---

---

### Description

Send a corpus to the google translate research API This function translates a the texts in a corpus by sending them to the google translate API.

### Usage

```
translate.corpus(corpus, targetlanguageString, textvar = "texts",
  languagevar = "language", key = NULL)
```

### Arguments

corpus	corpus to be translated
targetlanguageString	Language of the source text
languagevar	Language of the translated text

### Examples

```
## Not run:
translation <- translate(original, fr, de, key='insertkeyhere')

## End(Not run)
```

---

twitterSearch	<i>work-in-progress from-scratch interface to Twitter search API</i>
---------------	--

---

### Description

work-in-progress from-scratch interface to Twitter search API

### Usage

```
twitterSearch()
```

---

twitterStreamer	<i>work-in-progress interface to Twitter streaming API</i>
-----------------	--

---

**Description**

work-in-progress interface to Twitter streaming API

**Usage**

```
twitterStreamer()
```

---

twitterTerms	<i>make a corpus object from results of a twitter REST search</i>
--------------	---

---

**Description**

All of the attributes returned by the twitterR library call are included as attributes in the corpus. A oauth key is required, for further instruction about the oauth processs see: <https://dev.twitter.com/apps/new> and the twitterR documentation

**Usage**

```
twitterTerms(query, numResults = 50, key, cons_secret, token, access_secret)
```

**Arguments**

query	Search string for twitter
numResults	Number of results desired.
key	Number of results desired.
key	'your consumer key here'
cons_secret	'your consumer secret here'
token	'your access token here'
access_secret	'your access secret here'

**Examples**

```
## Not run:
twCorp <- twitterTerms('example', 10, key, cons_secret, token, access_secret)

## End(Not run)
```

---

uk2010immig

*Texts of UK 2010 manifestos on immigration*


---

### Description

Character vector of immigration-related sections from UK 2010 political party manifestos.

---

wordcloudDfm

*Plot a word cloud for a [dfm](#)*


---

### Description

plots a document as a wordcloud of its features

### Usage

```
wordcloudDfm(dfm, doc.index, ...)
```

### Arguments

dfm	document-feature matrix created in <a href="#">quanteda</a>
document	index of the document whose words will be plotted
...	additional arguments to pass to <a href="#">wordcloud</a>

### Value

None

### Author(s)

Kenneth Benoit

### Examples

```
data(iebudgets)
iebudgets2010 <- subset(iebudgets, year==2010)
wfm <- dfm(iebudgets2010, stopwords=TRUE)
wordcloudDfm(wfm, 1) # plot the finance minister's speech as a wordcloud
```

wordfishMCMC

*Bayesian-MCMC version of the "wordfish" Poisson scaling model***Description**

wordfishMCMC implements a flexible, Bayesian model estimated in JAGS using MCMC. It is based on the implementation of wordfish from the austin package. Options include specifying a model for alpha using document-level covariates, and partitioning the word parameters into different subsets, for instance, countries.

**Usage**

```
wordfishMCMC(dtm, dir = c(1, 2), control = list(sigma = 3, startparams =
  NULL), alphaModel = c("free", "logdoclength", "modelled"),
  alphaFormula = NULL, alphaData = NULL, wordPartition = NULL,
  betaPartition = FALSE, wordConstraints = NULL, verbose = TRUE,
  PoissonGLM = FALSE, nChains = 1, nAdapt = 100, nUpdate = 300,
  nSamples = 100, nThin = 1, ...)
```

**Arguments**

dtm	The document-term matrix. Ideally, documents form the rows of this matrix and words the columns, although it should be correctly coerced into the correct shape.
dir	A two-element vector, enforcing direction constraints on theta and beta, which ensure that $\theta[\text{dir}[1]] < \theta[\text{dir}[2]]$ . The elements of dir will index documents.
control	list specifies options for the estimation process. These are: tol, the proportional change in log likelihood sufficient to halt estimation, sigma the standard deviation for the beta prior in poisson form, and startparams a previously fitted wordfish model. verbose generates a running commentary during estimation. See <code>austin::wordfish</code> .
alphaModel	free means the $\alpha_i$ is entirely estimated; logdoclength means the alpha is predicted with an expected value equal to the log of the document length in words, similar to an offset in a Poisson model with variable exposure; modelled allows you to specify a formula and covariates for $\alpha_i$ using alphaFormula and alphaData.
alphaFormula	Model formula for hierarchical model predicting $\alpha_i$ .
alphaData	Data to form the model matrix for the hierarchical model predicting $\alpha_i$ .
wordPartition	A vector equal in length to the documents that specifies a unique value partitioning the word parameters. For example, alpha could be a Boolean variable for EU to indicate that a document came from a country outside the EU or inside the EU. Or, it could be a factor variable indicating the name of the country (as long as there are multiple documents per country). Internally, wordPartition is coerced to a factor. NULL indicates that no partitioning of the word-level parameters will take place (default).

betaPartition	Boolean indicating that the $\beta$ parameter should also be partitioned according to wordPartition.
wordConstraints	An index with a minimum length of 1, indicating which words will be set equal across the wordPartition factors. NULL if is.null(wordPartition) (default).
verbose	Turn this on for messages. Default is TRUE.
nChains	Number of chains to run in JAGS.
nAdapt	Adaptation iterations in JAGS.
nUpdate	Update iterations in JAGS.
nSamples	Number of posterior samples to draw in JAGS.
nThin	Thinning parameter for drawing posterior samples in JAGS.
PoissonGLM	Boolean denoting that the basic model should be estimated where $\log(\alpha)$ is $\sim \text{dflat}()$ as per The BUGS Book pp131-132
...	Additional arguments passed through.

**Value**

An augmented wordfish class object with additional stuff packed in. To be documented.

**Author(s)**

Kenneth Benoit

**Examples**

```
## Not run:
data(iebudgets)
# extract just the 2010 debates
iebudgets2010 <- corpus.subset(iebudgets, year==2010)

# create a document-term matrix and set the word margin to the columns
dtm <- create.fvm.corpus(iebudgets2010)
dtm <- wfm(t(dtm), word.margin=2)

# estimate the maximum likelihood wordfish model from austin
iebudgets2010_wordfish <- wordfish(dtm, dir=c(2,1))

# estimate the MCMC model, default values
iebudgets2010_wordfishMCMC <- wordfishMCMC(dtm, dir=c(2,1))

# compare the estimates of  $\theta_i$ 
plot(iebudgets2010_wordfish$theta, ibudgets2010_wordfishMCMC$theta)

# MCMC with a partition of the word parameters according to govt and opposition
# (FF and Greens were in government in during the debate over the 2010 budget)
# set the constraint on word partitioned parameters to be the same for "the" and "and"
iebudgets2010_wordfishMCMC_govtopp <-
  wordfishMCMC(dtm, dir=c(2,1),
```



```
wordPartition=(iebudgets2010$attribs$party=="FF" | iebudgets2010$attribs$party=="Green"),
  betaPartition=TRUE, wordConstraints=which(words(dtm)=="the"))

## End(Not run)
```

# Index

as.DocumentTermMatrix, 9  
austin, 19  
bigrams, 2  
collocations, 3  
corpus, 4  
countSyllables, 5  
describeTexts, 6  
dfm, 6, 7–10, 29, 33–35, 38  
dfm2ldaformat, 8  
dfm2tmformat, 9  
dfmSort, 9  
dfmTrim, 10  
directory, 11  
docnames, 11  
DocumentTermMatrix, 9  
docvar, 4  
file, 11  
filenames, 4  
flatten.dictionary, 12  
getRootFileNames, 13  
getTextDir, 13  
getTextDirGui, 14  
getTextFiles, 15  
getWordStat, 16  
getWordStatCSV, 16  
inaugCorpus, 16  
inaugTexts, 17  
kwic, 17  
kwic2, 18  
lda, 8  
lda.collapsed.gibbs.sampler, 8  
likelihood.test, 19  
MCMCirtPoisson1d, 19  
metacorporus, 21  
metacorporus<-, 22  
metacorporus<- (metacorporus), 21  
naiveBayesText, 22  
ngram, 2  
ngrams, 23  
predict.naivebayes, 24  
quanteda, 25  
quantedaRefresh, 25  
readWStatDict, 25  
sample.corpus, 26  
selectFeatures, 26  
sentenceSeg, 28  
settings, 28, 35  
settings<- (settings), 28  
simple triplet matrix, 9  
stopwords, 7  
stopwordsGet, 29  
stopwordsRemove, 29  
subset.corpus, 30  
summary.corpus, 31  
tagPos, 32  
texts, 32  
texts<- (texts), 32  
tf, 33  
tfidf, 34  
tokenise, 34  
tokenize, 2, 23  
translate, 35  
translate.corpus, 36  
twitterSearch, 36  
twitterStreamer, 37  
twitterTerms, 37  
uk2010immig, 38

wfm, [7](#)  
wordcloud, [38](#)  
wordcloudDfm, [38](#)  
wordfish, [19](#), [20](#)  
wordfishMCMC, [39](#)